Artificial Intelligence 175 (2011) 264–277

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

From answer set logic programming to circumscription via logic of GK

Fangzhen Lin [a,*], Yi Zhou [b]

[a] Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
[b] School of Computing and Mathematics, University of Western Sydney, Penrith South DC, NSW 1797, Australia

ARTICLE INFO

Article history:
Available online 3 April 2010

Keywords:
Logic programming
Answer set programming
Logic of GK
Circumscription
Nonmonotonic reasoning
Knowledge representation and reasoning

ABSTRACT

We first embed Pearce's equilibrium logic and Ferraris's propositional general logic programs in Lin and Shoham's logic of GK, a nonmonotonic modal logic that has been shown to include as special cases both Reiter's default logic in the propositional case and Moore's autoepistemic logic. From this embedding, we obtain a mapping from Ferraris's propositional general logic programs to circumscription, and show that this mapping can be used to check the strong equivalence between two propositional logic programs in classical logic. We also show that Ferraris's propositional general logic programs can be extended to the first-order case, and our mapping from Ferraris's propositional general logic programs to circumscription can be extended to the first-order case as well to provide a semantics for these first-order general logic programs.

Prologue

It gives us great pleasure to be able to contribute this work to this special issue of *Artificial Intelligence* in honor of John McCarthy. Like so many others, we have been influenced greatly by McCarthy and his work for as long as we have known AI. This particular work relates McCarthy's circumscription to several other nonmonotonic logics, and obviously could not have been done without McCarthy's pioneering work on nonmonotonic reasoning in general and circumscription in particular.

1. Introduction

Answer Set Programming (ASP) is a new paradigm of constraint-based programming based on logic programming with answer set semantics [9,13,17]. It started out with normal logic programs, which are programs that can have negation but not disjunction. Driven by the need of applications, various extensions have been proposed. These include disjunctive logic programs [5,6], nested expressions [7], cardinality and weight constraints [16], and others. Recently, Ferraris [2] proposed to view formulas in propositional logic as logic programs and showed that they include as special cases all the above mentioned classes of logic programs. In particular, Ferraris [2] provided a stable model semantics for these formulas using a transformation similar to the original Gelfond–Lifschitz transformation, and showed that this semantics coincides with Pearce's equilibrium logic [19].

In this paper, we show that this general stable model semantics can be embedded in Lin and Shoham's logic of GK (Grounded Knowledge) [11]. Besides showing the generality of Lin and Shoham's logic, which was proposed as a general logic for nonmonotonic reasoning, this embedding allows us to obtain a way to check in classical propositional logic whether any given two logic programs are strongly equivalent in almost the same way as in [12]. It also allows us to obtain a

* Corresponding author.
E-mail address: flin@cs.ust.hk (F. Lin).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.04.001

mapping from general logic programs to propositional circumscription in the same way as Lin and Shoham [11] did for mapping normal logic programs to circumscription. As it turned out, this mapping, when extended to first-order case, yields a semantics to first-order general logic programs that is similar to the one proposed recently by Ferraris et al. [4].

We first briefly review Lin and Shoham's logic of GK, Ferraris's general logic programs, and Pearce's equilibrium logic.

2. Logic of GK

The language of the logic of GK is a modal propositional language with two modal operators, K, for knowledge, and A, for assumption. Given a set *Atom* of atoms (also called variables or primitive propositions), formulas in the logic of GK are defined inductively below in BNF notation:

$$F ::= \bot \mid p \mid K(F) \mid A(F) \mid \neg F \mid F \wedge F \mid F \vee F \mid F \rightarrow F,$$

where $p \in Atom$, and $\bot$ is a constant standing for falsity. Formulas without modal operators are called *base formulas*.

The semantics of the logic of GK is defined through *Kripke interpretations*. A Kripke interpretation M is a tuple $\langle W, \pi, R_K, R_A, s\rangle$, where W is a nonempty set whose elements are called *possible worlds*, π is a function that maps a possible world to a truth assignment on *Atom*, R_K and R_A are binary relations over W representing the accessibility relations for K and A, respectively, and $s \in W$, called the *actual world* of M. The *satisfaction relation* $\models$ between a Kripke interpretation $M = \langle W, \pi, R_K, R_A, s\rangle$ and a formula F is defined inductively as follows:

- $M \not\models \bot$;
- $M \models p$ if $\pi(s)(p) = 1$, where $p \in Atom$;
- $M \models \neg F$ iff $M \not\models F$;
- $M \models F \wedge G$ iff $M \models F$ and $M \models G$;
- $M \models F \vee G$ iff $M \models F$ or $M \models G$;
- $M \models F \rightarrow G$ iff $M \not\models F$ or $M \models G$;
- $M \models K(F)$ iff $\langle W, \pi, R_K, R_A, w\rangle \models F$ for any $w \in W$, such that $(s, w) \in R_K$;
- $M \models A(F)$ iff $\langle W, \pi, R_K, R_A, w\rangle \models F$ for any $w \in W$, such that $(s, w) \in R_A$.

We say that a Kripke interpretation M is a *model* of a formula F if M satisfies F. In the following, given a Kripke interpretation M, we let

$$K(M) = \{F \mid F \text{ is a base formula and } M \models K(F)\},$$
$$A(M) = \{F \mid F \text{ is a base formula and } M \models A(F)\}.$$

Notice that $K(M)$ and $A(M)$ are always closed under classical logical entailment. In the following, for any set X of formulas, we let $Th(X)$ be the logical closure of X under classical logic.

Informally in GK, one assumes $A(M)$ and minimizes $K(M)$. When the assumed $A(M)$ turns out to be the same as the minimal $K(M)$, an equilibrium is reached, and the assumption is said to be "justified" or the knowledge is said to be "grounded". Formally, GK models are defined as follows.

Definition 2.1 *(GK models).* Given a formula F, a Kripke interpretation M is a *minimal model* of F if M is a model of F and there does not exist another model M_1 of F such that $A(M_1) = A(M)$ and $K(M_1) \subset K(M)$. We say that M is a *GK model*[1] if M is a minimal model of F and $K(M) = A(M)$.

Lin and Shoham showed that the logic of GK can be used to capture Reiter's default logic [20] and Moore's auto-epistemic logic [15]. As a consequence, normal logic programs under stable model semantics can be captured in the logic of GK as well. Specifically, they showed that a normal rule

$$r \leftarrow p_1, \ldots, p_n, \text{not}\, q_1, \ldots, \text{not}\, q_m$$

can be translated into the following sentence in the logic of GK:

$$Kp_1 \wedge \cdots \wedge Kp_n \wedge \neg Aq_1 \wedge \cdots \wedge \neg Aq_m \rightarrow Kr. \tag{1}$$

They also showed that this translation extends to disjunctive logic programs.

In this paper, we shall show that general logic programs proposed by Ferraris [2] can be captured in the logic of GK as well.

[1] In [11], GK models are called preferred models.

3. General logic programs

Given a set *Atom* of atoms, general logic programs [3] are formulas defined inductively below in BNF notation:

$$F ::= \bot \mid p \mid F \wedge F \mid F \vee F \mid F \rightarrow F,$$

where $p \in Atom$. Notice that there is no negation in the language. Instead, for any formula F, $\neg F$ is considered to be a shorthand for $F \rightarrow \bot$.

A set $X \subseteq Atom$ of atoms can be considered as a truth assignment in the straightforward way:

$$X \not\models \bot, \qquad X \models p \quad \text{iff} \quad p \in X,$$

and the usual definition for the logical connectives.

The stable models of a formula (general logic program) are defined by a modified extended Gelfond–Lifschitz transformation. Given a general logic program F, and a set X of atoms, the *reduct of F under X* [2], written F^X, is the formula obtained from F by replacing each maximal subformula that is not classically satisfied by X with $\bot$. Thus for example,

$$(\neg F)^X = \begin{cases} \top, & X \models \neg F, \\ \bot, & \text{otherwise.} \end{cases}$$

Now a set X of atoms is a stable model of a general logic program F if:

(i) $X \models F^X$;
(ii) there is no proper subset X_1 of X, such that $X_1 \models F^X$.

Example 3.1. Consider the following three general logic programs.

$$P = \neg p \rightarrow q,$$
$$Q = \neg p \vee p,$$
$$R = p \rightarrow \neg\neg p,$$

where p, q are atoms. The maximal subformula in P that is false under $\{q\}$ is p, thus $P^{\{q\}}$ is $\neg\bot \rightarrow q$, which is satisfied by $\{q\}$, but not by $\emptyset$. Therefore, $\{q\}$ is a stable model of P. On the other hand, $P^{\{p\}}$ is $\bot \rightarrow \bot$, which is satisfied by $\{p\}$ as well as its subset $\emptyset$. Therefore, $\{p\}$ is not a stable model of P. It can be seen that $\{q\}$ is the only stable model of P. Similarly, it can be shown that Q has two stable models, $\{p\}$ and $\emptyset$, and R has exactly one stable model $\emptyset$.

4. Pearce's equilibrium logic

Pearce's equilibrium logic [19] is based on the logic of here-and-there, a non-classical logic. Given a set *Atom* of atoms, formulas of *Atom* are exactly the same as in the case of general logic programs. Thus, negation in equilibrium logic is considered a shorthand as well.

The semantics of the logic of here-and-there is defined in terms of *HT-interpretations*, which are pairs $\langle X, Y\rangle$ of sets of atoms such that $X \subseteq Y$. The *HT satisfaction relation*[2] $\models$ between an HT-interpretation $\langle X, Y\rangle$ and a formula F is defined recursively as follows:

- For $p \in Atom$, $\langle X, Y\rangle \models p$ if $p \in X$;
- $\langle X, Y\rangle \not\models \bot$;
- $\langle X, Y\rangle \models F \wedge G$ if $\langle X, Y\rangle \models F$ and $\langle X, Y\rangle \models G$;
- $\langle X, Y\rangle \models F \vee G$ if $\langle X, Y\rangle \models F$ or $\langle X, Y\rangle \models G$;
- $\langle X, Y\rangle \models F \rightarrow G$ if
  (i) $\langle X, Y\rangle \not\models F$ or $\langle X, Y\rangle \models G$, and
  (ii) $Y \models F \rightarrow G$.

An HT interpretation $\langle X, Y\rangle$ is an *equilibrium model* of a formula F if $X = Y$, $\langle X, Y\rangle \models F$, and there is no proper subset X_1 of X, such that $\langle X_1, Y\rangle \models F$. Ferraris [2] showed that the stable models of a formula are essentially the same as its equilibrium models.

Theorem 1 *(Ferraris). Let X be a set of atoms and F a general logic program, X is a stable model of F iff $\langle X, X\rangle$ is an equilibrium model of F.*

[2] We overload $\models$, and use it to stand for satisfaction relations for modal logic, classical logic, and logic of here-and-there. Which one it stands for should be clear from the context.

5. From general logic programs and equilibrium logic to the logic of GK

In this section, we present a translation from a general logic program (also a formula in equilibrium logic) to a formula in the logic of GK, and show that under the translation, stable models (thus equilibrium models) coincide with GK models in the logic of GK.

Given a general logic program F, we define two formulas F_A and F_{GK} in the logic of GK as follows:

(1) F_A is obtained from F by simultaneously replacing each atom p by Ap.
(2) F_{GK} is defined inductively as follows:
- $\bot_{GK} = \bot$;
- For $p \in Atom$, $p_{GK} = Kp$;
- $(F \odot G)_{GK} = F_{GK} \odot G_{GK}$ ($\odot$ is $\wedge$ or $\vee$);
- $(F \to G)_{GK} = (F_{GK} \to G_{GK}) \wedge (F \to G)_A$.

It can be shown that for a normal logic program F, F_{GK} is equivalent to the translation by Lin and Shoham [11] given in Section 2 under

$$\bigwedge_{p \in Atom} Kp \to Ap, \tag{2}$$

and that for any formula W in the logic of GK, M is a GK model of W iff M is a GK model of $W \wedge (2)$.

This translation is also similar to the mapping from formulas in equilibrium logic to quantified boolean formulas given in [18]. We shall discuss this in more detail in a later section.

To illustrate, consider the three programs in Example 3.1. P_{GK} is

$$\big((\neg p)_{GK} \to q_{GK}\big) \wedge (\neg p \to q)_A,$$

which is

$$\big((\neg p_{GK} \wedge \neg p_A) \to Kq\big) \wedge (\neg p_A \to q_A),$$

which is

$$\big((\neg Kp \wedge \neg Ap) \to Kq\big) \wedge (\neg Ap \to Aq). \tag{3}$$

Now let M be a model of the above sentence. If $p \in A(M)$, then (3) holds no matter what $K(M)$ is, thus its minimal model is $K(M) = Th(\emptyset)$, so cannot be a GK model. Now if $p \notin A(M)$, then (3) is equivalent to $(\neg Kp \to Kq) \wedge Aq$. Thus $q \in A(M)$. Thus if M is a minimal model, then $K(M) = Th(\{q\})$. And if $A(M) = K(M)$, then M is a GK model. What we have shown here is that in any GK model M of (3), $K(M) = A(M) = Th(\{q\})$. The existence of such a model is apparent.

It can be similarly shown that Q_{GK} is equivalent to $Ap \to Kp$, and that M is a GK model of Q_{GK} iff $K(M) = A(M) = Th(\{p\})$ or $K(M) = A(M) = Th(\{\top\})$. And R_{GK} is equivalent to $Kp \to Ap$, and that M is a GK model of R_{GK} iff $K(M) = A(M) = Th(\{\top\})$. Thus for these three programs, their GK models correspond one-to-one with their stable models. In general, we have the following result. Proofs of the theorems are in Appendix A.

Theorem 2. *Let X be a set of atoms and F a general logic program. The following two statements are equivalent.*

(1) *X is a stable model of F.*
(2) *There is a GK model M of F_{GK} such that $K(M) = A(M) = Th(X)$.*

6. From general logic programs and equilibrium logic to circumscription

Given their mapping (1) from normal logic program to the logic of GK, Lin and Shoham [11] showed that stable model semantics for normal logic programs can be captured in circumscription [14] as follows. Given a set $Atom = \{p, q, \ldots\}$ of atoms, let $Atom' = \{p', q', \ldots\}$ be a new set of atoms. Given a normal logic program F, let $C(F)$ be the conjunction of the sentences:

$$p_1 \wedge \cdots \wedge p_n \wedge \neg q'_1 \wedge \cdots \wedge \neg q'_m \to r,$$

for each rule

$$r \leftarrow p_1, \ldots, p_n, \mathrm{not}\, q_1, \ldots, \mathrm{not}\, q_m$$

in F. Lin and Shoham [11] showed that X is a stable model of F iff $X \cup X'$ is a model of

$$\bigwedge_{p\in Atom}(p\leftrightarrow p')\wedge Circum\big(C(F);Atom\big),$$

where $Circum(W;Q)$ is the circumscription of the atoms in Q in W (with all other atoms fixed). Lin and Shoham also showed that this result can be extended to disjunctive logic programs. Using the same idea, we can capture the stable model semantics of general logic program and equilibrium logic in circumscription as well.

Let $Atom$ be a set of atoms. Again let $Atom' = \{p' \mid p \in Atom\}$ be a set of new atoms. For any $X \subseteq Atom$, let $X' = \{p' \mid p \in X\}$. Given any general logic program F in the language $Atom$, let $C(F)$ be the result obtained from F_{GK} by replacing every Kp in it by p and every Ap in it by p', for every $p \in Atom$.

Theorem 3. *For any general logic program F in the language Atom, any set $X \subseteq Atom$, the following two statements are equivalent*

(1) *There is a GK model M of F_{GK} such that $K(M) = A(M) = Th(X)$.*
(2) *$X \cup X'$ is a model of*

$$\bigwedge_{p\in Atom}(p\leftrightarrow p')\wedge Circum\big(C(F);Atom\big). \tag{4}$$

Interestingly, our translation $C(F)$ that embeds general logic programs and equilibrium logic in circumscription is exactly the same as the one by Pearce et al. [18] for embedding equilibrium logic in quantified boolean formulas. They showed that $\langle X, X\rangle$ is an equilibrium model of a formula F in equilibrium logic iff X' is a model of the following quantified boolean formula:

$$F'\wedge\neg\exists Atom\big(\big(Atom<Atom'\big)\wedge C(F)\big),$$

where F' is the formula obtained from F by replacing every atom p by p', and $Atom < Atom'$ stands for

$$\bigwedge_{p\in Atom}(p\to p')\wedge\neg\bigwedge_{p\in Atom}(p'\to p).$$

While propositional circumscription is also a quantified boolean formula, it is a well studied formalism. There are many known results about circumscription that we can use. Mapping logic programs to circumscription can help us better understand both formalisms.

Notice that the formula $\bigwedge_{p\in Atom}(p\leftrightarrow p')$ is equivalent to

$$\Big[\bigwedge_{p\in Atom}(p\to p')\Big]\wedge\Big[\bigwedge_{p\in Atom}(p'\to p)\Big].$$

Thus (4) is equivalent to

$$\Big[\bigwedge_{p\in Atom}(p\to p')\Big]\wedge\Big[\bigwedge_{p\in Atom}(p'\to p)\Big]\wedge Circum\big(C(F);Atom\big),$$

which is equivalent to

$$\Big[\bigwedge_{p\in Atom}(p'\to p)\Big]\wedge Circum\Big(C(F)\wedge\bigwedge_{p\in Atom}(p\to p');Atom\Big),$$

as the atoms (predicates) to be minimized occur only negatively in $\bigwedge_{p\in Atom}(p\to p')$. Putting the formula $\bigwedge_{p\in Atom}(p\to p')$ into the theory in the circumscription is good as it can be used to simplify $C(F)$.

Proposition 6.1. *If $\bigwedge_{p\in Atom}(p\to p')\models C(F)\leftrightarrow W$, then (4) is equivalent to*

$$\bigwedge_{p\in Atom}(p\leftrightarrow p')\wedge Circum(W;Atom).$$

Consider again the three programs in Example 3.1. For P, $C(P)$ is equivalent to $\neg p\wedge\neg p'\to q$, which is equivalent to $\neg p'\to q$ under $(p\to p')\wedge(q\to q')$. Thus for this program, (4) is equivalent to

$$\big(p\leftrightarrow p'\big)\wedge\big(q\leftrightarrow q'\big)\wedge Circum\big(\neg p'\to q;\{p,q\}\big),$$

which is equivalent to

$$(p \leftrightarrow p') \wedge (q \leftrightarrow q') \wedge \neg p \wedge (\neg p' \leftrightarrow q),$$

which has a unique model $\{q, q'\}$.

For $Q = \neg p \vee p$, $C(Q)$ is equivalent to $p' \rightarrow p$. Thus for this program, (4) is equivalent to

$$(p \leftrightarrow p') \wedge \mathit{Circum}(p' \rightarrow p; \{p\}),$$

which is equivalent to $p \leftrightarrow p'$, which has two models p and $\neg p$.

7. First-order general logic programs and circumscription

As in [11], we can extend the above mapping to the first-order case. First of all, we extend logic programs to the first-order case. Let L be a relational first-order language with equality, i.e. it has no proper functions. By an atom, we mean an atomic formula including equality atoms.

In the following, let Σ_{una} be the set of unique names assumptions on constants: $c_1 \neq c_2$ for any two distinct constants c_1 and c_2.

A first-order general logic program is a first-order sentence in the following set:

$$F ::= \bot \mid A \mid F \wedge F \mid F \vee F \mid F \rightarrow F \mid \forall x F \mid \exists x F,$$

where A is an atom, and x a variable. Again, for any general logic program F, $\neg F$ is considered to be a shorthand for $F \rightarrow \bot$.

Now let M be a finite model of Σ_{una} with domain D. Let σ be the mapping from constants to D under M. Clearly, for any distinct constants c_1 and c_2, $\sigma(c_1) \neq \sigma(c_2)$. We say that M is a stable model of a first-order general logic program F if $T(M)$, the set of ground facts true in M:

$$T(M) = \{P(\vec{u}) \mid P \text{ a predicate}, \vec{u} \in P^M\}$$

is a stable model of the general logic program F_M, the *grounding* of F on M obtained from F and M in two steps:

(1) First, replace every constant c in F by $\sigma(c)$, every subformula of the form $\forall x W$ in it by $\bigwedge_{u \in D} W(x/u)$, and every subformula of the form $\exists x W$ in it by $\bigvee_{u \in D} W(x/u)$, where $W(x/u)$ is the result of replacing every free occurrence of x in W by u. The order by which these subformulas are replaced does not matter.
(2) In the expression obtained by the first step, replace every equality atom $u = u$ by $\top$, and every $u = u'$ for distinct u and u' by $\bot$.

Example 7.1. Consider the following four programs:

$$F_1 = \exists x p(x) \wedge \exists x(\neg p(x) \rightarrow q),$$
$$F_2 = \exists x p(x) \wedge [(\exists x \neg p(x)) \rightarrow q],$$
$$F_3 = \exists x p(x) \wedge [\neg(\exists x p(x)) \rightarrow q],$$
$$F_4 = \exists x p(x) \wedge \forall x(\neg p(x) \rightarrow q).$$

Now consider a structure with two elements $\{a, b\}$. The grounding of the four programs on this domain are

$$(p(a) \vee p(b)) \wedge ((\neg p(a) \rightarrow q) \vee (\neg p(b) \rightarrow q)),$$
$$(p(a) \vee p(b)) \wedge ((\neg p(a) \vee \neg p(b)) \rightarrow q),$$
$$(p(a) \vee p(b)) \wedge (\neg(p(a) \vee p(b)) \rightarrow q),$$
$$(p(a) \vee p(b)) \wedge (\neg p(a) \rightarrow q) \wedge (\neg p(b) \rightarrow q),$$

respectively. So for this domain, F_1 and F_3 have the same stable models, $\{p(a)\}$ and $\{p(b)\}$, and F_2 and F_4 have the same stable models, $\{p(a), q\}$ and $\{p(b), q\}$. It is easy to see that this is the case for any given domain: F_1 and F_3 have the same stable models, and F_2 and F_4 have the same stable models.

We now show that the stable models of first-order general logic programs can be captured in circumscription as well.

Let Δ be the set of predicates in the language. Let Δ' be a set of new predicates, one for each P in Δ with the same arity and denoted by P'. Now given a first-order general logic program F, let $C(F)$ be the first-order formula defined inductively as follows.

- $C(\bot)$ is $\bot$.
- If W is an atomic formula, then $C(W)$ is W.

- $C(W_1 \odot W_2)$ is $C(W_1) \odot C(W_2)$, where $\odot \in \{\wedge, \vee\}$.
- $C(\forall x W)$ is $\forall x C(W)$, and $C(\exists x W)$ is $\exists x C(W)$.
- $C(W_1 \rightarrow W_2)$ is $(C(W_1) \rightarrow C(W_2)) \wedge (W_1' \rightarrow W_2')$, where W' is the result of replacing every predicate P in W by P'.

The stable models of F are then the models of the circumscription of all the predicates in Δ in $C(F)$, together with the following axiom

$$\bigwedge_{P \in \Delta} \forall \vec{x}\big(P(\vec{x}) \leftrightarrow P'(\vec{x})\big). \tag{5}$$

Theorem 4. *Let M be a finite model of Σ_{una}. M is a stable model of F iff M' is a model of*

$$Circum\big(C(F); \Delta\big) \wedge (5), \tag{6}$$

where M' is the conservative extension of M under (5).

Similar to Proposition 6.1, we have

Proposition 7.1. *If $\bigwedge_{P \in \Delta} \forall \vec{x}(P(\vec{x}) \rightarrow P'(\vec{x})) \models C(F) \leftrightarrow W$, then* (6) *is equivalent to*

$$Circum(W; \Delta) \wedge (5).$$

Example 7.2. Consider the programs in Example 7.1.

- $C(F_1)$ is

  $$\exists x p(x) \wedge \exists x\big[\big(\neg p(x) \wedge \neg p'(x) \rightarrow q\big) \wedge \big(\neg p'(x) \rightarrow q'\big)\big],$$

  which is equivalent to $\exists x p(x) \wedge (\neg \exists x p'(x) \rightarrow q)$ under $\forall x.p(x) \rightarrow p'(x)$. Therefore, under (5), $Circum(C(F_1), \{p, q\})$ is equivalent to

  $$\exists! x p(x) \wedge \big(\big(\neg \exists x p'(x)\big) \leftrightarrow q\big),$$

  thus equivalent to

  $$\exists! x p(x) \wedge \neg q,$$

  which can be considered to be the first-order semantics of F_1. If $D = \{a, b\}$, then there are exactly two models of this sentence, $\{p(a)\}$ and $\{p(b)\}$.
- $C(F_2)$ is

  $$\exists x p(x) \wedge \big(\exists x\big(\neg p(x) \wedge \neg p'(x)\big) \rightarrow q\big) \wedge \big(\exists x \neg p'(x) \rightarrow q'\big),$$

  which is equivalent to

  $$\exists x p(x) \wedge \big(\exists x \neg p'(x) \rightarrow q\big)$$

  under $\forall x.p(x) \rightarrow p'(x)$. Therefore, $Circum(C(F_2), \{p, q\})$ is equivalent to

  $$\exists! x p(x) \wedge \big(\exists x \neg p'(x) \leftrightarrow q\big),$$

  under (5), thus equivalent to

  $$\exists! x p(x) \wedge \big(\big(\exists x \neg p(x)\big) \leftrightarrow q\big),$$

  which can be considered to be the first-order semantics of F_2. If $D = \{a, b\}$, then there are exactly two models of this sentence, $\{p(a), q\}$ and $\{p(b), q\}$.
- $C(F_3)$ is

  $$\exists x p(x) \wedge \big(\neg \exists x p(x) \wedge \neg \exists x p'(x) \rightarrow q\big) \wedge \big(\neg \exists x p'(x) \rightarrow q'\big),$$

  which is equivalent to $C(F_1)$ under (5). Thus F_1 and F_3 are equivalent under our semantics.
- $C(F_4)$ is

  $$\exists x p(x) \wedge \forall x\big[\big(\neg p(x) \wedge \neg p'(x) \rightarrow q\big) \wedge \big(\neg p'(x) \rightarrow q'\big)\big],$$

  which is equivalent to $C(F_2)$. Thus F_2 and F_4 are equivalent under our semantics.

We have defined the stable model semantics for first-order logic programs by reducing them to propositional logic programs through grounding. For this to work, we need to assume a finite domain for otherwise a first-order sentence such as $\exists x p(x)$ cannot be replaced by a propositional sentence. We also need the unique names assumption in order to eliminate equality literals such as $a = b$ or $a \neq b$.

Theorem 4 shows that this stable model semantics for first-order logic programs can be captured by a simple formula (6) using circumscription. However, while the stable model semantics makes the unique names and finite models assumptions, the formula (6) is completely general and makes sense without these assumptions. Thus one could use this formula to define the stable model semantics of first-order logic programs in the general case. As it turned out, this will yield a semantics that is essentially the same as the one proposed recently by Ferraris et al. [4]. More precisely, given a first-order sentence F, Ferraris et al. proposed a second-order sentence as its stable model semantics, and showed that this second-order sentence is equivalent to (6) with all new predicates existentially quantified: $\exists \vec{p}\varphi(\vec{P}'/\vec{p})$, where φ is formula (6), $\vec{p}$ a tuple of predicate variables, one for each predicate P' in φ and with the same arity, and $\varphi(\vec{P}'/\vec{p})$ is the result of replacing each P' in φ by its corresponding variable in $\vec{p}$.

8. Strong equivalence

The notion of strong equivalence [8] is important in logic programming. For disjunctive logic programs, research by Lin and Chen [10] and Eiter et al. [1] show that interesting programs transformation rules can be designed based on the notion.

According to Ferraris and Lifschitz [3], two general logic programs F and G are said to be *strongly equivalent* if for any formula F_1 that contains an occurrence of F, F_1 has the same stable models as the formula obtained from it by replacing an occurrence of F by G. They showed that for any F and G, they are strongly equivalent iff F and G are equivalent in the logic here-and-there.

As it turns out, our mapping from equilibrium logic to logic of GK also embeds logic of here-and-there to modal logic. Thus the problem of deciding whether two programs are strongly equivalent can be reduced to checking whether certain modal logic formulas are valid, and that, because of the special format of these modal formulas, can in turn be reduced to checking whether certain propositional formulas are tautologies.

Theorem 5. *Let F be a formula in equilibrium logic, X and Y two sets of atoms such that $X \subseteq Y$, and M a Kripke interpretation such that $K(M) = Th(X)$ and $A(M) = Th(Y)$. We have that $\langle X, Y\rangle \models F$ iff $M \models F_{GK}$.*

Theorem 6. *Let F and G be two general logic programs. The following conditions are equivalent.*

1. *F and G are strongly equivalent.*
2. $\bigwedge_{p\in Atom}(Kp \to Ap) \models (F \leftrightarrow G)_{GK}$.
3. $\bigwedge_{p\in Atom}(Kp \to Ap) \models F_{GK} \leftrightarrow G_{GK}$.
4. $\bigwedge_{p\in Atom}(p \to p') \models C(F \leftrightarrow G)$.
5. $\bigwedge_{p\in Atom}(p \to p') \models C(F) \leftrightarrow C(G)$.

Corollary 7. *The problem of deciding whether two general logic programs are strongly equivalent is co-NP complete.*

9. Conclusion

We showed that the logic of GK proposed by Lin and Shoham is flexible enough to handle stable model semantics of general logic programs. Because of this, the stable model semantics of general logic programs can also be formulated in circumscription, in both propositional and first-order cases. For future work, we plan to make use of the expressive power of GK in other applications.

Acknowledgements

We thank Vladimir Lifschitz for stimulating discussions on topics related to this paper, and for his helpful comments on earlier versions of this paper. This work was supported in part by HK RGC under grant HKUST6170/04E, and by China NSFC under grants 60703095 and 90718009.

Appendix A. Proofs of theorems

A.1. Proof of Theorem 2

Lemma 1. *Let F be a general logic program; X and Y two sets of atoms such that $X \subseteq Y$ and M a Kripke interpretation such that $K(M) = Th(X)$, $A(M) = Th(Y)$. We have that $X \models F^Y$ iff $M \models F_{GK}$.*

Proof. Let F and G be two general logic programs, $X \subseteq Y$ two sets of atoms. As shown in [3], $(F \wedge G)^X$ is equivalent to $F^X \wedge G^X$, and $(F \vee G)^X$ equivalent to $F^X \vee G^X$. Thus by induction on the structure of F, it's easy to see that $Y \models F$ iff $M \models F_A$, where M is as in the proposition.

We now prove that $X \models F^Y$ iff $M \models F_{GK}$ by induction on the structure of F.

- If F is $\bot$, then this assertion holds obviously.
- If F is an atom p, then p_{GK} is Kp. $X \models p^Y$ iff $p \in X$. On the other hand, $M \models Kp$ iff $p \in X$ too. Therefore, $X \models p^Y$ iff $M \models p_{GK}$.
- If F is $G \wedge H$, $X \models (G \wedge H)^Y$ iff $X \models G^Y$ and $X \models H^Y$. By induction hypothesis, this holds iff $M \models G_{GK}$ and $M \models H_{GK}$, which is equivalent to $M \models (G \wedge H)_{GK}$.
- If F is $G \vee H$, $X \models (G \vee H)^Y$ iff $X \models G^Y$ or $X \models H^Y$. By induction hypothesis, this holds iff $M \models G_{GK}$ or $M \models H_{GK}$, which is equivalent to $M \models (G \vee H)_{GK}$.
- Finally, if F is $G \to H$. $X \models (G \to H)^Y$ iff $Y \models G \to H$ and $X \models G^Y \to H^Y$ iff (i) $Y \models G \to H$ and (ii) $X \not\models G^Y$ or $X \models H^Y$. By induction hypothesis, this holds iff (i) $M \models (G \to H)_A$ and (ii) $M \not\models G_{GK}$ or $M \models H_{GK}$ iff $M \models (G \to H)_{GK}$.

This completes the induction proof. □

Theorem 2. *Let X be a set of atoms and F a general logic program. The following two statements are equivalent.*

(1) *X is a stable model of F.*
(2) *There is a GK model M of F_{GK} such that $K(M) = A(M) = Th(X)$.*

Proof. (1) $\Rightarrow$ (2): Suppose that X is a stable model of F. We can construct a Kripke interpretation M such that $K(M) = A(M) = Th(X)$. We now prove that M is a GK model of F_{GK}. Firstly, we have $X \models F^X$ since X is a stable model of F. By Lemma 1, $M \models F_{GK}$. We only need to prove M is a minimal model. We prove it by contradiction. Assume that M is not a minimal model of F_{GK}, then there exists another model M_1 of F_{GK} such that $A(M_1) = A(M) = Th(X)$ and $K(M_1) \subset K(M)$. We construct another Kripke interpretation M_2 such that $A(M_2) = Th(X)$ and $K(M_2) = Th(X_1)$, where $X_1 = \{p \mid M_1 \models Kp, p \in Atom\}$. For any general logic program G, by induction on the structure of G, it's easy to see that M_2 is a model of G_{GK} iff M_1 is a model of G_{GK}. Therefore, M_2 is a model of F_{GK}. By Lemma 1, $X_1 \models F^X$. Moreover, $X_1 \subset X$. This shows that X is not a stable model of F, a contradiction. Hence, M is a GK model of F_{GK}.

(1) $\Leftarrow$ (2): Suppose that there is a GK model M of F_{GK} such that $K(M) = A(M) = Th(X)$. By Lemma 1, $X \models F^X$. There is no proper subset X_1 of X such that $X_1 \models F^X$. Otherwise, we can construct a Kripke interpretation M_1 such that $K(M_1) = Th(X_1)$ and $A(M_1) = Th(X)$. By Lemma 1, M_1 is also a model of F_{GK}. Moreover $K(M_1) \subset K(M)$; $A(M_1) = A(M)$. This shows that M is not a GK model of F_{GK}, a contradiction. Hence, X is a stable model of F. □

A.2. Proof of Theorem 3

Notice that $C(F)$ can also be defined recursively as follows.

- $C(\bot)$ is $\bot$.
- If F is an atom, then $C(F)$ is F.
- $C(F_1 \odot F_2)$ is $C(F_1) \odot C(F_2)$, where $\odot$ is $\wedge$ or $\vee$.
- $C(F_1 \to F_2)$ is $(C(F_1) \to C(F_2)) \wedge (F_1' \to F_2')$, where F' is the expression obtained from F by replacing every atom p by p'.

Lemma 2. *Let F be a general logic program, X and Y two sets of atoms, and M a Kripke interpretation such that $K(M) = Th(X)$, $A(M) = Th(Y)$. We have that $X \cup Y'$ is a model of $C(F)$ iff $M \models F_{GK}$.*

Proof. We prove this by induction on the structure of F.

- If F is $\bot$, then this assertion holds obviously.
- If F is an atom p, then p_{GK} is Kp, $C(F)$ is p. $X \cup Y' \models p$ iff $p \in X$. On the other hand, $M \models Kp$ iff $p \in X$ too. Therefore, $X \cup Y' \models C(p)$ iff $M \models p_{GK}$.
- If F is $G \wedge H$, $X \cup Y' \models C(G \wedge H)$ iff $X \cup Y' \models C(G)$ and $X \cup Y' \models C(H)$. By induction hypothesis, this holds iff $M \models G_{GK}$ and $M \models H_{GK}$, which is equivalent to $M \models (G \wedge H)_{GK}$.
- If F is $G \vee H$, $X \cup Y' \models C(G \vee H)$ iff $X \cup Y' \models C(G)$ or $X \cup Y' \models C(H)$. By induction hypothesis, this holds iff $M \models G_{GK}$ or $M \models H_{GK}$, which is equivalent to $M \models (G \vee H)_{GK}$.
- Finally, if F is $G \to H$. $X \cup Y' \models C(G \to H)$ iff $X \cup Y' \models (C(G) \to C(H)) \wedge (G \to H)'$ iff $X \cup Y' \models (G \to H)'$ and $X \cup Y' \models C(G) \to C(H)$ iff (i) $X \cup Y' \models (G \to H)'$ and (ii) $X \cup Y' \not\models C(G)$ or $X \cup Y' \models C(H)$. Notice that $X \cup Y' \models (G \to H)'$ iff

$Y' \models (G \to H)'$ iff $Y \models G \to H$ iff $M \models (G \to H)_A$. By induction hypothesis, $X \cup Y' \models C(F)$ iff (i) $M \models (G \to H)_A$ and (ii) $M \not\models G_{GK}$ or $M \models H_{GK}$ iff $M \models (G \to H)_{GK}$.

This completes the induction proof. □

Theorem 3. *For any general logic program F in the language Atom, any set $X \subseteq Atom$, the following two statements are equivalent*

(1) *There is a GK model M of F_{GK} such that $K(M) = A(M) = Th(X)$.*
(2) *$X \cup X'$ is a model of*

$$\bigwedge_{p \in Atom} (p \leftrightarrow p') \wedge Circum\big(C(F); Atom\big). \tag{4}$$

Proof. (1) ⇒ (2): Suppose that M is a GK model of F_{GK} and $K(M) = A(M) = Th(X)$. Then $M \models F_{GK}$, by Lemma 2, $X \cup X'$ is a model of $C(F)$. We now show that for any set of atoms such that $X_1 \subset X$, $X_1 \cup X'$ is not a model of $C(F)$. Suppose otherwise, $X_1 \cup X'$ is a model of $C(F)$. Construct a GK interpretation M_1 such that $K(M_1) = Th(X_1)$ and $A(M_1) = Th(X)$. By Lemma 2, $M_1 \models F_{GK}$. It's clear that $K(M_1) \subset K(M)$, $A(M_1) = A(M)$. This shows that M is not a GK model of F_{GK}, a contradiction. Therefore, $X \cup X'$ is a model of $Circum(C(F); Atom)$. Of course, $X \cup X'$ is a model of $\bigwedge_{p \in Atom}(p \leftrightarrow p')$. This shows that statement (1) implies statement (2).

(1) ⇐ (2): Suppose that $X \cup X'$ is a model of $Circum(C(F); Atom)$. Thus $X \cup X'$ is a model of $C(F)$ and for every proper subset X_1 of X, $X_1 \cup X'$ is not a model of $C(F)$. Let M be a Kripke interpretation such that $K(M) = A(M) = Th(X)$. We now prove that M is a GK model of F_{GK}. According to Lemma 2, $M \models F_{GK}$. We only need to prove that M is a minimal model. We prove it by contradiction. Assume that M is not a minimal model of F_{GK}. Then there exists another model M_1 of F_{GK} such that $A(M_1) = A(M) = Th(X)$ and $K(M_1) \subset K(M)$. Construct another Kripke interpretation M_2 such that $A(M_2) = Th(X)$ and $K(M_2) = Th(X_1)$, where $X_1 = \{p \mid M_1 \models Kp, p \in Atom\}$. For any general logic program G, by induction on the structure of G, M_2 is a model of G_{GK} iff M_1 is a model of G_{GK}. Therefore, M_2 is a model of F_{GK}. By Lemma 2, $X_1 \cup X' \models C(F)$. Moreover, $X_1 \subset X$. $X \cup X'$ is not a model of $Circum(C(F); Atom)$, a contradiction. Hence, M is a GK model of F_{GK}. This shows that statement (2) implies statement (1). □

A.3. Proof of Theorem 4

Let L be a first order language with equality but without proper functions. Let C be the set of constants in L, and Δ the set of predicates in L. Let $\Delta' = \{P' \mid P \in \Delta\}$ be a set of new predicates, and L' the extension of L by the new predicates in Δ'.

A first-order structure M' of L' is said to be a conservative extension of a first-order structure M of L if M' and M have the same domain, and they interpret all symbols in L the same. Suppose that M' is a conservative extension of M. It is easy to see that if M is a model of Σ_{una}, then M' is also a model of Σ_{una}, and that for every sentence F, F_M and $F_{M'}$ are the same.

Let F be a first order general logic program in L, and M a finite model of Σ_{una}. We remark that F_M can also be defined recursively as follows (recall that σ is the mapping from constants in L to the domain of M under M).

- $\bot_M$ is $\bot$.
- If F is $a = b$, where a and b are two constants, then F_M is $\top$ when $\sigma(a)$ is the same as $\sigma(b)$, otherwise F_M is $\bot$.
- If F is an atomic formula $P(\vec{t})$, where $\vec{t}$ is a vector of constants, then F_M is $P(\sigma(\vec{t}))$, where $\sigma(\vec{t})$ is $\langle \sigma(t_1), \ldots, \sigma(t_n)\rangle$ when $\vec{t}$ is $\langle t_1, \ldots, t_n\rangle$.
- $(G \odot H)_M$ is $G_M \odot H_M$, where $\odot$ is $\wedge$, $\vee$ or $\to$.
- $(\exists x F)_M$ is $\bigvee_{u \in D} F(x/u)_M$.
- $(\forall x F)_M$ is $\bigwedge_{u \in D} F(x/u)_M$.

Thus given a first order general logic program F in L, F_M is a propositional formula in $A(M)$, the set of ground atoms in M:

$$A(M) = \big\{P(\vec{u}) \;\big|\; P \text{ an } n\text{-ary predicate}, \vec{u} \in D^n, D \text{ the domain of } M\big\},$$

and $C(F_M)$ is a propositional formula in $A(M) \cup A(M)'$.

Let M be a first-order structure. Recall that $T(M)$ is the set of ground atoms true in M:

$$T(M) = \big\{P(\vec{u}) \;\big|\; P \text{ a predicate}, \vec{u} \in P^M\big\}.$$

Lemma 3. *Let M be a finite model of Σ_{una}, and F a first order general logic program. We have that M is a model of F iff $T(M)$ is a model of F_M.*

Proof. We prove it by induction on the structure of F.

- If F is $\bot$, then this assertion holds obviously.
- If F is $a = b$, where a, b are constants, then F_M is $\top$ when a and b are the same; F_M is $\bot$ when a and b are two distinct constants. On the other hand, since M is a model of Σ_{una}, M is a model of F iff a is the same with b. Therefore this assertion holds.
- If F is an atomic formula $p(\vec{t})$, where $\vec{t}$ is a vector of constants, then F_M is $p(\sigma(\vec{t}))$. M is a model of F iff $\sigma(\vec{t}) \in p^M$ iff $p(\sigma(\vec{t})) \in T(M)$ iff $T(M)$ is a model of F_M.
- If F is $G \wedge H$, then F_M is $G_M \wedge H_M$. M is a model of F iff M is a model of $G \wedge H$ iff M is a model of G and F is a model of H iff $T(M)$ is a model of G_M and $T(M)$ is a model of H_M iff $T(M)$ is a model of $G_M \wedge H_M$ iff $T(M)$ is a model of F_M.
- If F is $G \vee H$, then F_M is $G_M \vee H_M$. M is a model of F iff M is a model of $G \vee H$ iff M is a model of G or F is a model of H iff $T(M)$ is a model of G_M or $T(M)$ is a model of H_M iff $T(M)$ is a model of $G_M \vee H_M$ iff $T(M)$ is a model of F_M.
- If F is $G \to H$, then F_M is $G_M \to H_M$. M is a model of F iff M is a model of $G \to H$ iff M is not a model of G or F is a model of H iff $T(M)$ is not a model of G_M or $T(M)$ is a model of H_M iff $T(M)$ is a model of $G_M \to H_M$ iff $T(M)$ is a model of F_M.
- If F is $\exists x G$, then F_M is $\bigvee_{u \in D}(G(x/u))_M$. M is a model of F iff there exists $u \in D$ such that M is a model of $G(x/u)$ iff there exists $u \in D$ such that $T(M)$ is a model of $(G(x/u))_M$ iff $T(M)$ is a model of $\bigvee_{u \in D}(G(x/u))_M$ iff $T(M)$ is a model of F_M.
- If F is $\forall x G$, then F_M is $\bigwedge_{u \in D}(G(x/u))_M$. M is a model of F iff for all $u \in D$, M is a model of $G(x/u)$ iff for all $u \in D$, $T(M)$ is a model of $(G(x/u))_M$ iff $T(M)$ is a model of $\bigwedge_{u \in D}(G(x/u))_M$ iff $T(M)$ is a model of F. □

Lemma 4. *Let M be a finite model of Σ_{una}, and F a first order general logic program. We have that $C(F_M)$ is equivalent to $C(F)_M$.*

Proof. We prove it by induction on the structure of F.

- If F is $\bot$, this assertion holds obviously.
- F is $a = b$, where a, b are constants. If a is the same with b, then both $C(F_M)$ and $(C(F))_M$ are $\top$, otherwise both of them are $\bot$.
- F is an atomic formula $p(\vec{t})$. Then F_M is $p(\sigma(\vec{t}))$, $C(F_M)$ is also $p(\sigma(\vec{t}))$. On the other hand, $C(F)$ is $p(\vec{t})$, $(C(F))_M$ is $p(\sigma(\vec{t}))$ too.
- F is $G \wedge H$. Then $C(F_M)$ is equivalent to $C(G_M) \wedge C(H_M)$, which is equivalent to $(C(G))_M \wedge (C(H))_M$, which is equivalent to $(C(G) \wedge C(H))_M$, which is equivalent to $(C(G \wedge H))_M$, which is $(C(F))_M$.
- F is $G \vee H$. Then $C(F_M)$ is equivalent to $C(G_M) \vee C(H_M)$, which is equivalent to $(C(G))_M \vee (C(H))_M$, which is equivalent to $(C(G) \vee C(H))_M$, which is equivalent to $(C(G \vee H))_M$, which is $(C(F))_M$.
- F is $G \to H$. Then $C(F_M)$ is $C(G_M \to H_M)$, which is equivalent to $(C(G_M) \to C(H_M)) \wedge (G'_M \to H'_M)$, which is equivalent to $(C(G)_M \to C(H)_M) \wedge (G'_M \to H'_M)$. On the other hand, $C(F)_M$ is equivalent to $((C(G) \to C(H)) \wedge (G' \to H'))_M$, which is equivalent to $(C(G)_M \to C(H)_M) \wedge (G'_M \to H'_M)$. Moreover, it's easy to see that for any first order sentence H and any structure M, H'_M is the same as $(H_M)'$. Hence, this assertion holds.
- If F is $\exists x G$, then F_M is $\bigvee_{u \in D}(G(x/u))_M$. $C(F_M)$ is $\bigvee_{u \in D} C((G(x/u))_M)$, which is equivalent to $\bigvee_{u \in D}(C(G(x/u))_M)$. On the other hand, $C(F)$ is $\exists x C(G)$, $(C(F))_M$ is $\bigvee_{u \in D}(C(G)(x/u))_M$. Moreover, by induction on the structure, it's easy to see that for any first order formula H, $C(H(x/u))$ is equivalent to $C(H)(x/u)$. Hence, this assertion holds.
- If F is $\forall x G$, then F_M is $\bigwedge_{u \in D}(G(x/u))_M$. $C(F_M)$ is $\bigwedge_{u \in D} C((G(x/u))_M)$, which is equivalent to $\bigwedge_{u \in D}(C(G(x/u))_M)$. On the other hand, $C(F)$ is $\forall x C(G)$, $(C(F))_M$ is $\bigwedge_{u \in D}(C(G)(x/u))_M$. Similarly, this assertion holds. □

Lemma 5. *Let M be a finite model of Σ_{una}, M' a conservative extension of M under (5), and F a first order sentence. Then $T(M) \cup T(M)'$ is a model of $C(F_M)$ iff M' is a model of $C(F)$.*

Proof. Since M' is a conservative extension of M under (5), $T(M') = T(M) \cup T(M)'$.

M' is a model of $C(F)$
iff
$T(M')$ is a model of $C(F)_{M'}$ (Lemma 3)
iff
$T(M) \cup T(M)'$ is a model of $C(F)_M$
iff
$T(M) \cup T(M)'$ is a model of $C(F_M)$ (Lemma 4). □

Let M and M^* be two first order structures on L' and Δ the set of predicates in L. We say that M^* is less than M on Δ, written by $M^* \subset_\Delta M$, if and only if:

(1) M and M^* have the same domain D.
(2) M and M^* map every constant c in C into the same element in D.
(3) For each $p' \in \Delta'$, $\vec{u} \in p'^{M^*}$ iff $\vec{u} \in p'^{M}$.
(4) For all $p \in \Delta$, if $\vec{u} \in p^{M^*}$, then $\vec{u} \in p^{M}$.
(5) There is a $p \in \Delta$ and some $\vec{u}$ such that $\vec{u} \in p^{M}$ and $\vec{u} \notin p_i^{M^*}$.

Thus M is a model of $Circum(F; \Delta)$ if and only if M is a model of F and there is no $M^* \subset_\Delta M$ such that M^* is also a model of F.

Lemma 6. *Let M be a finite model of Σ_{una}, M' a conservative extension of M under (5), F a first order sentence, and $S \subseteq A(M)$. We have that*

(1) *$S \subset T(M)$ iff there is M^* such that $M^* \subset_\Delta M'$ and $T(M^*) = S \cup T(M)'$.*
(2) *For any M^* such that $M^* \subset_\Delta M'$ and $T(M^*) = S \cup T(M)'$, $S \cup T(M)'$ is a model of $C(F_M)$ iff M^* is a model of $C(F)$.*

Proof. (1) is obvious. Proof of (2):

M^* is a model of $C(F)$
iff
$T(M^*)$ is a model of $C(F)_{M^*}$ (Lemma 3)
iff
$S \cup T(M)'$ is a model of $C(F)_{M^*}$
iff
$S \cup T(M)'$ is a model of $C(F)_M$ (M and M^* have the same domain and interpret all constants the same)
iff
$S \cup T(M)'$ is a model of $C(F_M)$ (Lemma 4). □

Theorem 4. *Let M be a finite model of Σ_{una}. M is a stable model of F iff M' is a model of*

$$Circum\big(C(F); \Delta\big) \wedge (5), \tag{6}$$

where M' is the conservative extension of M under (5).

Proof. M is a stable model of F
iff
$T(M)$ is a stable model of F_M (by the definition)
iff
$T(M) \cup T(M)'$ is a model of $\bigwedge_{p \in A(M)}(p \leftrightarrow p') \wedge Circum(C(F_M); A(M))$ (by Theorem 3)
iff

- $T(M) \cup T(M)'$ is a model of $\bigwedge_{p \in A(M)}(p \leftrightarrow p')$;
- $T(M) \cup T(M)'$ is a model of $C(F_M)$;
- for any $S \subseteq A(M)$, if $S \subset T(M)$, then $S \cup T(M)'$ is not a model of $C(F_M)$

iff

- M' is a model of (5);
- M' is a model of $C(F)$ (by Lemma 5);
- for any first order structure M^*, if $M^* \subset_\Delta M'$, then M^* is not a model of $C(F)$ (by Lemma 6)

iff M' is a model of (6). □

A.4. Proof of Theorems 5 and 6

Theorem 5. *Let F be a general logic program; X and Y two sets of atoms such that $X \subseteq Y$ and M a Kripke interpretation such that $K(M) = Th(X)$, $A(M) = Th(Y)$. We have that $\langle X, Y\rangle \models F$ iff $M \models F_{GK}$.*

Proof. As stated in [2], $\langle X, Y\rangle \models F$ iff $X \models F^Y$. According to Lemma 1, $X \models F^Y$ iff $M \models F_{GK}$, therefore, this assertion holds. □

Theorem 6. *Let F and G be two general logic programs. The following statements are equivalent.*

1. *F and G are strongly equivalent.*

2. $\bigwedge_{p\in Atom}(Kp \to Ap) \models (F \leftrightarrow G)_{GK}$.
3. $\bigwedge_{p\in Atom}(Kp \to Ap) \models F_{GK} \leftrightarrow G_{GK}$.
4. $\bigwedge_{p\in Atom}(p \to p') \models C(F \leftrightarrow G)$.
5. $\bigwedge_{p\in Atom}(p \to p') \models C(F) \leftrightarrow C(G)$.

Proof. $1 \Rightarrow 2$: Let M be a model of $\bigwedge_{p\in Atom}(Kp \to Ap)$. Construct a Kripke interpretation M_1 such that $K(M_1) = Th(X)$ and $A(M_1) = Th(Y)$, where $X = \{p \mid M \models Kp, p \in Atom\}$; $Y = \{p \mid M \models Ap, p \in Atom\}$. It's clear that $M_1 \models (F \leftrightarrow G)_{GK}$ iff $M \models (F \leftrightarrow G)_{GK}$. Since M is a model of $\bigwedge_{p\in Atom}(Kp \to Ap)$, $X \subseteq Y$. Therefore $\langle X, Y\rangle$ is an HT-interpretation. Since F and G are assumed to be strongly equivalent, they are equivalent in the logic of here-and-there. Thus $\langle X, Y\rangle \models F \leftrightarrow G$. Then by Theorem 5, $M_1 \models (F \leftrightarrow G)_{GK}$. Therefore, $M \models (F \leftrightarrow G)_{GK}$. This shows that $\bigwedge_{p\in Atom}(Kp \to Ap) \models (F \leftrightarrow G)_{GK}$.

$2 \Rightarrow 3$: $(F \leftrightarrow G)_{GK}$ is $((F \to G) \wedge (G \to F))_{GK}$, which is equivalent to $(F \to G)_{GK} \wedge (G \to F)_{GK}$, which is equivalent to

$$(F_{GK} \to G_{GK}) \wedge (F_A \to G_A) \wedge (G_{GK} \to F_{GK}) \wedge (G_A \to F_A),$$

which is equivalent to $(F_{GK} \leftrightarrow G_{GK}) \wedge (F_A \leftrightarrow G_A)$. This shows that $(F \leftrightarrow G)_{GK} \models F_{GK} \leftrightarrow G_{GK}$. Hence, this assertion holds.

$3 \Rightarrow 1$: We first show that if $\bigwedge_{p\in Atom}(Kp \to Ap) \models F_{GK} \leftrightarrow G_{GK}$, then F is classically equivalent to G. Suppose otherwise, without loss of generality, X is a model of F but not a model of G. By the definition of reduction, it's clear that X is a model of F^X. Construct a Kripke interpretation M such that $K(M) = A(M) = Th(X)$. By Lemma 1, M is a model of F_{GK}. Thus, M is also a model of G_{GK}. Again by Lemma 1, X is a model of G^X. Thus, X is a model of G, a contradiction. Hence, if $\bigwedge_{p\in Atom}(Kp \to Ap) \models F_{GK} \leftrightarrow G_{GK}$, then F is classically equivalent to G. Therefore, F_A is equivalent to G_A.

Suppose that H is a general logic program and it contains an occurrence of F, H_1 is the general logic program obtained from H by replacing an occurrence of F in H by G. We now show that H has the same set of stable models as H_1. By 3, $\bigwedge_{p\in Atom}(Kp \to Ap) \models F_{GK} \leftrightarrow G_{GK}$. Therefore, by induction on the structure of H, it's easy to see that $\bigwedge_{p\in Atom}(Kp \to Ap) \models H_{GK} \leftrightarrow (H_1)_{GK}$. Thus, H_{GK} and $(H_1)_{GK}$ has the same set of GK models. By Theorem 2, H has the same set of stable models as H_1.

$2 \Rightarrow 4$: Suppose $\bigwedge_{p\in Atom}(Kp \to Ap) \models (F \leftrightarrow G)_{GK}$. Now, suppose that X and Y are two sets of atoms and $X \cup Y'$ is a model of $\bigwedge_{p\in Atom}(p \to p')$. Therefore, $X \subseteq Y$. Construct a Kripke interpretation M such that $K(M) = Th(X)$ and $A(M) = Th(Y)$. We have that M is a model of $\bigwedge_{p\in Atom}(Kp \to Ap)$. By 2, it's also a model of $(F \leftrightarrow G)_{GK}$. By Lemma 2, $X \cup Y'$ is a model of $C(F \leftrightarrow G)$. This shows that $\bigwedge_{p\in Atom}(p \to p') \models C(F \leftrightarrow G)$.

$4 \Rightarrow 5$: Similar to the proof of $2 \Rightarrow 3$, $C(F \leftrightarrow G) \models C(F) \leftrightarrow C(G)$. Hence, this assertion holds.

$5 \Rightarrow 1$: Similar to the proof of $3 \Rightarrow 1$. □

Let F be a propositional formula constructed with atoms and basic connectives ($\bot$, $\wedge$, $\vee$ and $\to$). Let $Length(F)$ be the total number of atoms and $\bot$ occurring in F.

Lemma 7.

$$Length\big(C(F)\big) \leqslant \big(Length(F)+1\big)\big(Length(F)+2\big)/2 - 2.$$

Proof. We prove this assertion by induction on the structure of F.

- If F is $\bot$, then $Length(F) = Length(C(F)) = 1$. This assertion holds.
- If F is an atom p, then $Length(F) = Length(C(F)) = 1$. This assertion holds as well.
- If F is $G \wedge H$ or $G \vee H$, then $Length(C(F)) = Length(C(G)) + Length(C(H)) \leqslant (Length(G)+1)(Length(G)+2)/2 - 2 + (Length(H)+1)(Length(H)+2)/2 - 2 \leqslant (Length(G)+Length(H)+1)(Length(G)+Length(H)+2)/2 - 2$ (which is equivalent to $0 \leqslant Length(G)Length(H)+1) \leqslant (Length(F)+1)(Length(F)+2)/2 - 2$.
- Finally, if F is $G \to H$, then $Length(C(F)) = Length(C(G)) + Length(C(H)) + Length(F) \leqslant (Length(G)+1)(Length(G)+2)/2 - 2 + (Length(H)+1)(Length(H)+2)/2 - 2 + Length(G) + Length(H) \leqslant (Length(G)+Length(H)+1)(Length(G)+Length(H)+2)/2 - 2$ (which is equivalent to $0 \leqslant (Length(G)-1)(Length(H)-1)) \leqslant (Length(F)+1)(Length(F)+2)/2 - 2$.

This completes the proof. □

Corollary 7. *The problem of deciding whether two general logic programs are strongly equivalent is co-NP complete.*

Proof. This assertion follows straightforwardly from Theorem 6 and Lemma 7. □

References

[1] T. Eiter, M. Fink, H. Tompits, P. Traxler, S. Woltran, Replacements in non-ground answer-set programming, in: KR'2006, 2006, pp. 340–351.
[2] P. Ferraris, Answer sets for propositional theories, in: LPNMR, 2005, pp. 119–131.

[3] P. Ferraris, V. Lifschitz, Mathematical foundations of answer set programming, in: We Will Show Them! (1), 2005, pp. 615–664.
[4] P. Ferraris, J. Lee, V. Lifschitz, A new perspective on stable models, in: Proceedings of IJCAI'07, 2007, pp. 372–379.
[5] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Generation Computing 9 (1991) 365–385.
[6] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, F. Scarcello, The DLV system for knowledge representation and reasoning, ACM Transactions on Computational Logic 7 (3) (2006) 499–562.
[7] V. Lifschitz, L.R. Tang, H. Turner, Nested expressions in logic programs, Annals of Mathematics and Artificial Intelligence 25 (3–4) (1999) 369–389.
[8] V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, ACM Transactions on Computational Logic 2 (4) (2001) 526–541.
[9] V. Lifschitz, Action languages, answer sets and planning, in: K.R. Apt, V.W. Marek, M. Truszczynski, D.S. Warren (Eds.), The Logic Programming Paradigm: A 25-Year Perspective, Springer-Verlag, 1999.
[10] F. Lin, Y. Chen, Discovering classes of strongly equivalent logic programs, in: Proc. of IJCAI'95, 2005, pp. 516–521.
[11] F. Lin, Y. Shoham, A logic of knowledge and justified assumptions, Artificial Intelligence 57 (1992) 271–289.
[12] F. Lin, Reducing strong equivalence of logic programs to entailment in classical propositional logic, in: Proc. of KR'02, 2002, pp. 170–176.
[13] V.W. Marek, M. Truszczynski, Stable logic programming – an alternative logic programming paradigm, in: K.R. Apt, V.W. Marek, M. Truszczynski, D.S. Warren (Eds.), The Logic Programming Paradigm: A 25-Year Perspective, Springer-Verlag, 1999.
[14] J. McCarthy, Applications of circumscription to formalizing commonsense knowledge, Artificial Intelligence 28 (1986) 89–118.
[15] R. Moore, Semantical considerations on nonmonotonic logic, Artificial Intelligence 25 (1) (1985) 75–94.
[16] I. Niemelä, P. Simons, Extending the smodels system with cardinality and weight constraints, in: Jack Minker (Ed.), Logic-Based Artificial Intelligence, Kluwer Academic Publishers, 2000, pp. 491–521.
[17] I. Niemelä, Logic programs with stable model semantics as a constraint programming paradigm, Annals of Mathematics and Artificial Intelligence 25 (3–4) (1999) 241–273.
[18] D. Pearce, H. Tompits, S. Woltran, Encodings for equilibrium logic and logic programs with nested expressions, in: Proc. EPIA-01, 2001, pp. 306–320.
[19] D. Pearce, A new logical characterisation of stable models and answer sets, in: Non-Monotonic Extensions of Logic Programming, 1997, pp. 57–70.
[20] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1980) 81–132.